Implementing the Markov Chain and Inferring About the Restored Sequence

A Discussion of

"Accurate Restoration of DNA Sequences"

by G. A. Churchill

George Casella*
Cornell University

Christian P. Robert†
Ura CNRS 1378, Université de Rouen

November 22, 1993

BU-1224-M

1 Introduction

The article by Professor Churchill provides a wonderful introduction to this fascinating subject. We heartily congratulate him. In this discussion we would

*Research Supported by NSF Grant No. DMS 9305547 and by the joint NSF-CNRS Grant No. INT 9216784. This is paper BU-1224-M in the Biometrics Unit, Cornell University, Ithaca, NY 14853.

†This research was performed while Professor Robert was visiting Cornell University, supported by the joint NSF-CNRS Grant No. INT 9216784

like to examine three points in detail, with the joint goal of assessing the ability of the present methodology to produce a usable inferential procedure. First, we look in detail at implementing the Markov chain, both in computing the necessary distributions and generating the required random variables. Second, we outline a procedure for constructing a confidence set for the restored clone sequence. Lastly, we discuss the feasibility of implementing the algorithms.

2 Model and Notation

The Markov Chain model, as given by Churchill in Section 3.1.4, is

$$\begin{aligned} \{s,\Gamma\}^{(j)} &\sim \{s,\Gamma\}|F, A^{(j-1)}, \theta^{(j-1)} \\ \theta^{(j)} &\sim \theta|F, \{s,\Gamma\}^{(j)}, A^{(j-1)} \\ A^{(j)} &\sim A|F, s^{(j)}, \theta^{(j)} \end{aligned} \tag{1}$$

where

$$\begin{aligned} \{s,\Gamma\}^{(j)} = s^{*(j)} &= \text{the restored clone sequence together with alignment information;} \\ F &= \{f_1, \cdots, f_m\} \text{ are the m fragments;} \\ A^{(j)} &= \text{the assembly information for the fragments;} \\ \theta^{(j)} &= \{(\tau,\ \mu), (\lambda,\ \pi_I(\cdot)),\ \pi_R(\cdot|\cdot)\} \text{ are underlying parameters.} \end{aligned}$$

Together, F and A result in an $n_A \times m$ matrix $X = \{x_{ij}\}$, the assembled fragment set. This is the alignment of the fragments F according to the information in A. A few clarifying remarks are in order.

i) The difference between s and $\{s,\Gamma\}$ is subtle, but important. The clone sequence s, of length n_S, takes its values in $\{A, C, G, T\}$. The sequence $\{s,\Gamma\}$,

of length n_A, takes its values in $\{-, A, C, G, T, \mathcal{A}^k, k = 2, 3, \cdots\}$ where $\mathcal{A}^k$ is made of all k-tuples of the $\{A, C, G, T\}$ alphabet. For example, we may have

$$s^* = \{s, \Gamma\} = |A|C| - |G^A|T| - |$$

of length $n_A = 6$, where the corresponding s is $ACGAT$ of length $n_S = 5$. So $\{s, \Gamma\}$ is the clone sequence together with the alignment information, which is what is generated, while s is the inferred clone sequence, the object of interest.

ii) When generating the alignments $\vec{\alpha_1}, \cdots, \vec{\alpha_m}$ we, in fact, generate a new Γ for the clone sequence s. Thus, we may write the third step in the Markov chain (1) as

$$\{A, \Gamma^{(j)}\} \sim \{A, \Gamma\}|F, s^{(j)}, \theta^{(j)} \tag{2}$$

We could update the Γ part of $\{s, \Gamma\}^{(j)}$ at this point, or use only the $A^{(j)}$ from this generation.

iii) The groupings of parameters in θ reflect their purpose: (τ, μ) govern the beginning and ending of the copying process; $(\lambda, \pi_I(\cdot))$ govern the insertion process; and $\pi_R(\cdot|\cdot)$ governs the replacement process. In this discussion we will focus on a special case of π_I and π_R, but the mechanics of generalization are straightforward. We consider the special case

$$\begin{aligned}
\pi_I(\cdot) &= \frac{1}{4} \\
\pi_R(b_1|b_2) &= p_S/4, \quad b_1 \in \mathcal{B},\ b_2 \in \mathcal{A},\ b_1 \neq b_2 \\
\pi_R(-|b) &= p_D, \quad b \in \mathcal{A} \\
\pi_R(b|b) &= 1 - p_S - p_D, \quad b \in \mathcal{A}.
\end{aligned}$$

This is the case where all insertions are equally likely, as are all substitutions. To make these distinctions more clear, the following small example may be helpful.

Example: Suppose we have fragments $f_1 = AC$, $f_2 = GAT$, $f_3 = ACAT$.

• At step i, the generated clone sequence and alignment is

$$\{s, \Gamma\}_i = |A||G|| - |C|C^G|T| \tag{3}$$

(Note the C^G element is in $\mathcal{A}^2$.) The resulting generated clone sequence is s = AGCCGT. (The mechanics of mapping $\{s, \Gamma\}$ to s are to delete dashes and string out insertions.)

• Given s and the three fragments, we generate three alignments $\vec{\alpha_1}$, $\vec{\alpha_2}$ and $\vec{\alpha_3}$

$\{f_1, \vec{\alpha_1}\}$						$\{f_2, \vec{\alpha_2}\}$						$\{f_3, \vec{\alpha_3}\}$					
A	G	C	C	G	T	A	G	C	C	G	T	A	G	C	C	G	T
A	-	C	-	-	-	-	-	-	-	G^A	T	A	-	C	-	A	T

(4)

These alignments result in $\{A, \Gamma\}_i$, the assembly and alignment information, and the assembled fragments matrix X_i, where

$$\{A, \Gamma\}_i = \begin{bmatrix} A & - & C & - & - & - & - \\ - & - & - & - & G & A & T \\ A & - & C & - & - & A & T \end{bmatrix},$$

$$X_i = \begin{bmatrix} A & C & - & - & - \\ - & - & G & A & T \\ A & C & - & A & T \end{bmatrix} \tag{5}$$

(The mechanics of $\{A, \Gamma\} \rightarrow X$ are to delete columns that contain only dashes.)

• Given the assembled fragments matrix X_i, we generate a new realization

of $\{s, \Gamma\}$. The generation is done on a column by column basis:

$$X_i = \begin{array}{c} \begin{array}{ccccc} A^T & C & - & A & T \end{array} \\ \left[\begin{array}{ccccc} A & C & - & - & - \\ - & - & G & A & T \\ A & C & - & A & T \end{array}\right] \end{array}$$

yielding the new $\{s, \Gamma\} = |A^T|C| - |A|T|$. When run to equilibrium, the output from the Gibbs sampler is a sample $\{s, \Gamma\}_i$, $i = 1, \cdots, k$ from the marginal distribution. In the next section we look at all of the steps of the chain in detail, and examine exactly how the needed densities are calculated and the random variables are generated.

3 Calculation and Generation of the Chain

There are three parts to the generation of the Markov Chain in (1), and each part presents its own difficulties. We will treat them in order.

It first will be useful to discuss various groupings of parameters and statistics, and the forms of these are easiest to work with. The collection of fragments, F, represent the data (or, in EM algorithm terms, the "incomplete" data). A good choice of the "complete" data is $X = \{F, A\}$, the assembled fragments matrix. With this data, the parameter (clone) vector $s^* = \{s, \Gamma\}$ is most straightforward to work with. Thus, we implement the Markov Chain (1) as

$$\begin{aligned} s^{*(j)} &\sim s^* | X^{(j-1)}, \theta^{(j-1)} \\ \theta^{(j)} &\sim \theta | X^{(j-1)}, s^{*(j-1)} \\ A^{(j)} &\sim A | F, s^{(j)}, \theta^{(j)}. \end{aligned} \tag{6}$$

Note that in the third step of (6) we only condition on $s^{(j)}$ from $s^{*(j)} = \{s, \Gamma\}^{(j)}$, and we actually generate a new Γ, which is discarded. The assembly $A^{(j)}$ is then used to update $X^{(j-1)}$ to $X^{(j)}$. In deriving the necessary posterior distributions we will work with the complete data likelihoods expressed in terms of X, as given by Churchill in Section 3.2. This strategy is easier to implement than working with the likelihoods based on F

3.1 The $\{s, \Gamma\}$ distribution

Recalling that $X = \{F, A\}$, the desired distribution for the first part of the Markov chain in (1) is $\{s, \Gamma\}|\ X, \theta$. The matrix X is $n_A \times m$, and the vector $\{s, \Gamma\}$ is $n_A \times 1$. An element of $\{s, \Gamma\}$, say $\{s, \Gamma\}_i$, takes its values in $\{-\cup \mathcal{A}^k,\ k = 1, 2, \cdots\}$ and we will write either $\{s, \Gamma\}_i = -$ or $\{s, \Gamma\}_i = b \times b_{k-1}$. It is important to separate the first element in the k-tuple, as this is the only base for which the corresponding x_{ij} imparts any information. Now, using the specification of the prior and sampling distribution given by Churchill in Section 3.2, and using the fact that the columns of X are assumed independent, straightforward calculation yields the posterior distribution

$$\begin{aligned} P\left(\{s, \Gamma\}_i = b \times b_{k-1} | \mathbf{x}_i, \theta\right) &\propto \left(1 - p_S - p_D\right)^{\delta_i(b)} \left(\frac{1}{3} p_S + p_D\right)^{m - \delta_i(b)} \\ &\quad \times \frac{(1 - \eta_0)(1 - \eta_1)\eta_1{}^{k-1}}{4^k} \qquad (7) \\ P\left(\{s, \Gamma\}_i = - | \mathbf{x}_i, \theta\right) &\propto \left(\frac{1}{4} \lambda^{T_i}\right) (1 - \lambda)^{m - T_i} \eta_0, \end{aligned}$$

where $\mathbf{x}_i = i^{th}$ column of X, $\delta_i(b) = \sum_{j=1}^{m} I(x_{ij} = b)$ and $T_i = \delta_i(A) + \delta_i(C) + \delta_i(G) + \delta_i(T)$. The normalizing constant is simple to compute, being a sum of the geometric series, hence available in closed form. Finally, the full posterior

of $\{s, \Gamma\}$ is, by independence,

$$P\left(\{s,\Gamma\}|\mathbf{X},\theta\right) = \prod_{i=1}^{n_A} P\left(\{s,\Gamma\}_i|\mathbf{x_i},\theta\right) \tag{8}$$

Generation of $\{s, \Gamma\}_i$ is, perhaps, most easily accomplished by first generating Γ_i (the "depth" of the element $\{s, \Gamma\}_i$), and then generating $\{s, \Gamma\}_i | \Gamma_i$. We do this using, from (7),

$$\begin{aligned} P\left(\Gamma_i = 0|\mathbf{x_i},\theta\right) &\propto \left(\frac{1}{4}\lambda\right)^{T_i}(1-\lambda)^{m-T_i}\eta_0 \\ P\left(\Gamma_i = k|\mathbf{x_i},\theta\right) &\propto (1-\eta_0)\,(1-\eta_1)^{\frac{\eta_1^{k-1}}{4^k}} \\ &\quad \times \sum_{b\in\{A,C,G,T\}} (1-p_S-p_D)^{\delta_i(b)}\left(\frac{1}{3}p_S+p_D\right)^{m-\delta_i(b)} \end{aligned}$$

and

$$\begin{aligned} P\left(\{s,\Gamma\}_i = -|\Gamma_i = 0,\mathbf{x_i},\theta\right) &= 1 \\ P\left(\{s,\Gamma\}_i = b\times b_{k-1}|\Gamma_i = k,\mathbf{x_i},\theta\right) &\propto (1-p_S-p_D)^{\delta_i(b)}\left(\frac{1}{3}p_S+p_D\right)^{m-\delta_i(b)}\frac{1}{4^{k-1}}. \end{aligned}$$

Thus, Γ_i is generated using a geometric distribution, and then $\{s, \Gamma\}_i$ is a straightforward discrete generation. The necessary normalizing constants can also be easily calculated.

3.2 The Distribution of θ

The parameter vector $\theta = \{(\tau, \mu), \lambda, (p_S, p_D)\}$ plays no role in the ultimate inference on s, but provides an essential intermediate step in the model. Fortunately, calculation of the posterior distribution and generation of random variables is straightforward. We again start from the likelihood in terms of X, given by Churchill in Section 3.2. Recalling that $\{s, \Gamma\} = s^*$ and $\{F, A\} = X$, the posterior distribution of θ in (1) is

$$\pi(\theta|F,\{s,\Gamma\},A) = \pi(\theta|X,s^*) \propto P(X|s^*,\theta)P(s^*|\theta)\pi(\theta) \tag{9}$$

and, since the elements of X are independent given s^*, we can write

$$\pi(\theta|X, s^*) \propto \prod_{i=1}^{n_A} \prod_{j=1}^{m} P\left(x_{ij}|s_i^*, \theta\right) P\left(s_i^*|\theta\right) \pi(\theta). \tag{10}$$

Using the distributions given by Churchill in Section 3.2, and defining

$$\begin{aligned}
\delta_{ij}(u, v) &= \begin{cases} 1 & \text{if } x_{ij} = u \text{ and } s_i^* = v \\ 0 & \text{otherwise,} \end{cases} \\
N_i^{(B)} &= \text{number of } \phi s \text{ in } x_{ij} \text{ before copying begins,} \\
N_i^{(E)} &= \text{number of } \phi s \text{ in } x_{ij} \text{ before copying ends,}
\end{aligned}$$

we have

$$\begin{aligned}
\pi(\theta|X, s^*) \quad \propto \quad & [(1-\lambda)\eta_0]^{\sum_{ij} \delta_{ij}(-,-)} \prod_{k=1}^{\infty} \left[p_D(1-\eta_0)(1-\eta_1)\eta_1^{k-1}\right]^{\sum_{ij} \delta_{ij}(-,b_k)} \\
\times \quad & \prod_{l=1}^{4} [\lambda\gamma_0/4]^{\sum_{ij} \delta_{ij}(b_1^{(l)},-)} \qquad (11) \\
\times \quad & \prod_{l=1}^{4} \prod_{k=1}^{\infty} \left[\frac{1-p_S-p_D}{4}(1-\eta_0)(1-\eta_1)\eta^{k-2_1}\right]^{\sum_{ij} \delta_{ij}(b_1^{(l)}, b_1^{(l)} b_{k-1})} \\
\times \quad & \prod_{l=1}^{4} \prod_{l'=1 l \neq l'}^{4} \left[\frac{p_S}{4}(1-\eta_0)(1-\eta_1)\eta_1^{k-2}\right]^{\sum_{ij} \delta_{ij}(b_1^{(l)}, b_1^{(l')} b_{k-1})} \\
\times \quad & \tau\mu(1-\tau)^{\sum_i N_i^{(B)}} (1-\mu)^{\sum_i N_i^{(E)}} \times \pi(\theta)
\end{aligned}$$

Although the products in (11) are infinite products, in practice they contain only n_A terms. This is because if s_i^* has depth d, then $\delta(u, \text{v})$ is zero unless v also has depth d. If we then take $\pi(\theta)$ to be a product of a Dirichlet distribution on p_S and p_D, and independent beta distributions on λ, τ, μ, η_0 and η_1, the posterior distribution is again a product of a Dirichlet and independent beta distributions. Note that our $\delta_{ij}(\cdot, \cdot)$ notation is a more explicit form of Churchill's notation in Section 3.4. The δ_{ij} notation involves X and s^*, while the t_{ab} notation

involves X and s. Of course, we could have written $\pi(\theta|X,\ s^*)$ in terms of different component distributions, in particular using the likelihood on F given by Churchill in Section 3.1.3. This would lead to

$$\pi(\theta|F,\{s,\Gamma\},A) \propto P(F,\vec{\alpha}|s,\theta)P(s,\theta) = P(F|\vec{\alpha},s,\theta)P(\vec{\alpha}|s,\theta)P(s|\theta)\pi(\theta)$$

where we have used the fact that $\{F,\vec{\alpha}\} = \{F,A,\Gamma\}$ However, this form of the posterior distribution seems much more difficult to work with. In particular, the distribution $P(s|\theta)$ is quite involved.

3.3 The Alignment Distribution

The third part of the Markov chain (1), the distribution of A_i, poses the most difficulties in implementation. Although we do not have a simple expression for the distribution of $A|F,s,\theta$, we can describe an algorithm for the generation of A. Fortunately, this is all that we need. Using the independence of the fragments, the assembly is generated on a row$\times$ row basis, with row i only dependent on fragment f_i. For a given f_i, we must generate an alignment $\vec{\alpha}_i$, a row vector of n_{α_i} elements, with each element taking values in $\{0,1,2\}$, as described by Churchill in Section 3.1.2. Taken together, we get a row vector $\{f_i,\vec{\alpha}_i\}$, of length n_{α_i}, describing the alignment of fragment f_i with the clone sequence s (as shown in (4)). Note that each vector $\{f_i,\vec{\alpha}_i\}$ may have different lengths. The m vectors $\{f_i,\vec{\alpha}_i\}$, $i = 1,\cdots,m$ are then aligned together to form $\{A,\Gamma\}$ (see (5)), where gaps are inserted in each $\{f_i,\vec{\alpha}_i\}$ to correspond to gaps in s generated from $\{f_j,\vec{\alpha}_j\}$, $j \neq i$. Finally, the assembled fragments matrix X is obtained by deleting from $\{A,\Gamma\}$ all columns that contain only gaps. Therefore, the generation of X, the desired variable, follows directly from generation of each

vector $\vec{\alpha}_i$. To generate $\vec{\alpha}_i$, we use the algorithm described in detail by Churchill in Sections 3.3.2 and 3.3.3.

4 Inference About the Clone Sequence

Once a sample of clone sequences $s^{(1)}, \cdots, s^{(k)}$ has been obtained from the Gibbs sampler, we can then combine this information into a composite "confidence" clone sequence. This would actually be a confidence set (or, more precisely, a Bayesian credible set) on the true sequence. Of course, it would be desirable to construct such a confidence set in an optimal manner, but it is not clear to us what the optimality criterion should be, or if an optimal construction is even feasible. We therefore content ourselves with presenting a method that leads to a usable confidence set, but almost certainly not an optimal set. We also point out some strategies for optimization. First, from the sample $s^{(1)}, \cdots, s^{(k)}$, identify the sequence s^M (the "modal" sequence) with the highest posterior probability,

$$P(s^M|F) = \max_i P(s^{(i)}|F).$$

Next, for a chosen distance function d, calculate d_i = distance between $s^{(i)}$ and s^M. Assume, without loss of generality, that $d_1 \leq d_2 \leq \cdots \leq d_{k-1}$. Then find the smallest value of k^* such that, for a specified confidence value 1–α,

$$P\left(\{s^M, s^{(1)}, \ldots, s^{(k^*)}\}|F\right) \geq 1 - \alpha$$

and take $s^C = \{s^M, s^{(1)}, \ldots, s^{(k^*)}\}$ as a 1-α confidence set. The set s^C is itself a clone sequence with some ambiguous characters. For example, we might have a sequence s^C of length 8 given by

$$s^C = A|*|*|G|C|C \text{ or } T|*|T|$$

for a 95% confidence set. It is hoped that the elements of s^C will be less ambiguous where fragment alignment is unequivocal, and more ambiguous near the ends of the clone, where fragment alignment is more problematic. The probability calculations required to implement the algorithm are all straightforward, and follow directly from the Gibbs sampler. For each sequence $s^{(i)}$ we have

$$\begin{aligned} P\left(s^{(i)}|F\right) &\approx \frac{1}{k}\sum_{j=1}^{k} P\left(s^{(i)}|F, A, \Gamma_j, \theta_j\right) \\ &= \frac{1}{k}\sum_{j=1}^{k} \frac{P\left(\{s^{(i)}, \Gamma_j\}|F, A_j, \theta_j\right)}{P\left(\Gamma_j|F, A_j, \theta_j\right)} \\ &= \frac{1}{k}\sum_{j=1}^{k} \frac{P\left(\{s^{(i)}, \Gamma_j\}|X_j, \theta_j\right)}{P\left(\Gamma_j|X_j, \theta_j\right)} \end{aligned}$$

where X_j is the j-th assembled fragments matrix. The probabilities can now be calculated from the formulas in Section 3.1. The distance measure d can take many forms, but an optimal form is not known. It might be reasonable to try a "0–1" distance, where $d(s^{(i)}, s^{(j)})$ is the number of non-matching bases. (A minor complication is caused by the fact that the alignment of $s^{(i)}$ and $s^{(j)}$ is not well defined. This can be accommodated by calculating a minimum or maximum distance.) Other choices for the distance function can be based on a probabilistic weighting (giving higher weight to bases with higher probability) or methods based on the scoring schemes of Karlin et al. (1990).

5 Practical Aspects of Implementing the Chain

Each of the three steps of the Markov chain result in a straightforward algorithm for generating random variables from a posterior distribution. Thus, the Markov chain can be run to its stable distribution, and a sample of clone sequences $s^{(1)}$,

$\cdots$, $s^{(k)}$ from $\pi(s|\ \mathrm{F})$ can be drawn. However, moving from theoretical calculations to practical implementation can often be extremely difficult, especially when a problem is of the magnitude of this one. We now address some of these difficulties.

5.1 Assessing the Computation Time

The first difficulty encountered in implementing the Markov chain is the massive amount of computing that is necessary. At this time, this computing roadblock makes the Monte Carlo Markov chain (MCMC) algorithm formally available but, in practice, impossible to implement.

To make this point clearer, consider a sequence of $10,000$ bases and $1,000$ extracted fragments of average length 100 bases, sizes that would occur in practice. For each fragment, the computation of the probability matrix of §3.3 is of an order of $3 \times 100 \times 1,000 = 3 \times 10^5$. The simulation of A requires computations of the order of 3×10^8 (unless a faster method for simulating the alignment vectors, α_i, is discovered). Compared with this impressive amount of computation, the simulations of $\{s, \Gamma\}$ and of θ take a negligible amount of computation time. This implies that a single step of the Gibbs sampler requires 3×10^8 units of basic CPU time. Although 3×10^8 is quite a manageable amount of operations for today's computers, in particular in parallel setups with each fragment being run separately, we must also take into account the fact that the Gibbs sampler requires a very large number of iterations to be efficient in this particular setup. In fact, although the sequence s takes values on a finite state space, $\{A, C, G, T\}^{\mathcal{G}}$, the cardinality of the state space is properly appalling since it reaches $4^{\mathcal{G}} \simeq 10^{6,000}$ when $\mathcal{G} = 10,000$.

The finiteness of the state space guarantees proper convergence of the Gibbs sampler and most MCMC methods. It also ensures geometric convergence of the Markov chain of the s's (Tierney, 1991) and even of the other chains generated simultaneously (Robert, 1993; Diebolt and Robert, 1993, 1994). But this theoretical reassurance is worth very little in practice since it does not give any hint at the actual speed of the algorithm. In fact, similar setups with huge finite state spaces, like those of the Ising model examined in Gelman and Rubin (1992), have pointed out the difficulty of attaining stationarity, as well as the dependency on startup conditions. So the theoretical picture associated with the Gibbs sampler in this setting is quite clear: since we can simulate any value of the sequence s from any previous value, the Markov chain $s^{(j)}$ produced by the Gibbs sampler is irreducible and aperiodic, and therefore recurrent and ergodic. There exists a single stationary distribution, the true posterior distribution of s, and convergence to this distribution is geometric and even φ-mixing (Billingsley, 1968). However, the practical behaviour of the Gibbs sampler in such setups is pretty much unknown. For one thing, the (conditional) probabilities of most states of the sequence must be negligible, but the width of the state space is such that the significant values cannot be identified easily and, more importantly, that it will usually require a considerable number of iterations to move from one mode of the posterior distribution to another, i.e., to explore thoroughly the posterior surface. It is then quite likely that "apparent stabilization" phenomena, as those exhibited in Gelman and Rubin (1992), can occur.

The assembly line inertia is also likely to be quite high under the Gibbs sampler perturbations. Consider the case of a single fragment f_i being misplaced.

The number of iterations required for the actual position of f_i to occur can be quite high if the present location of f_i has an influencial effect on the corresponding sequence, i.e., if the present position of f_i is then much more probable under the simulated s than its true position. This difficuly is obviously magnified by the number of possible starting values for a given fragment, roughly $(2\mathcal{G})^{|f|}$ if the sequence is of length $\mathcal{G}$, and, evidently, by the number of fragments. We are again facing huge numbers, of the order of magnitude of 10^{400}. It could be interesting to make use of the present deterministic techniques of sequencing to derive a starting point for the algorithm or, on the contrary, to see how many iterations of the Gibbs sampler are required to see this sequence (or a close modification) appear. Starting with existing techniques makes the Gibbs sampler appear as another type of *simulated annealling*, i.e. of a technique where deterministic solutions are randomly perturbated to see whether more interesting solutions exist in their neighbourhood.

5.2 Alternative Algorithms

A computing alternative to the algorithm proposed by Churchill is to call for another MCMC device. Although Gibbs sampling is often the most natural and the simplest choice of MCMC algorithm in a Bayesian setup, and it certainly is in this case, Gibbs sampling can suffer from slow convergence properties in complex settings due to difficulty in escaping local modes of the posterior distribution. More "energetic" perturbations, like those induced by the *Metropolis-Hasting* algorithm (see Tierney, 1991), may be more appropriate. In fact, while exploring more thoroughly the parameter space, this algorithm simultaneously reduces the computing time required for each iteration. If we select the distribution

to be simulated from –called the "working" distribution– for the fragments to be simple enough both for simulation and for analytic form, the computation time can be cut down considerably. For example, for a fragment f, while the simulation time is of order $|f|$, the probability weight involved in the Metropolis acceptance step is also of that order since we only need to follow the path corresponding to f in the matrix of Churchill's Figure 4.

Along with this saving of 10^3 orders of magnitude and the apparent simplicity of the Metropolis algorithm, there are also disadvantages to the Metropolis approach. In fact, the chain generated by the Metropolis algorithm can remain at a certain spot for very long time if the Metropolis weights are too small to induce a change. However, this criticism applies to every implementation of this algorithm, but does really not have any strong theoretical backing. Replacement of a "true" simulation step in a Gibbs algorithm by a Metropolis approximation retains the same convergence properties as the original chain.

In practice, the Metropolis algorithm is often less likely to get stuck than a regular Gibbs sampler because of the larger scale of random perturbations it involves. For Churchill's setting of DNA sequences, we can even suggest some approaches to the implementation of the Metropolis approximation. In fact, the working distribution can be chosen as a random walk perturbation of the previous alignment of each fragment. The parameters of this random walk have to be chosen with care since, if they induce too small a variation in the chain, potential trapping states may appear for the simulated chain (although they are impossible theoretically). Alternatively, if the corresponding variance is too large, the chain will be too perturbated to achieve any visible stationarity. We therefore suggest a preliminary tuning of these parameters in the first (1,000?

10,000?) iterations of the algorithm in order to achieve a range of rejection between 30% and 70% as in Muller (1992) or Besag and Mengersen (1993). Once this stable mode of perturbation is reached, the Metropolis algorithm can then be modified into an hybrid Metropolis algorithm, with two different magnitudes of perturbation. In fact, as the algorithm approaches the mode of the posterior distribution, large variations between simulations actually slow convergence. Therefore, we suggest the use of occasional shake-ups in the simulation, with intermediate moderate perturbations, in order to preserve global modes but also to ensure a 'uniform' coverage of the parameter space.

5.3 The Number of Paths

A last, somewhat more technical, remark. Gary Churchill enquired during his talk about the number of paths through a p by n matrix when the only possible moves from (i,j) are to $(i,j+1)$, $(i+1,j)$ and $(i+1,j+1)$. This number is actually

$$\sum_{k=0}^{n\wedge p} \binom{n+p-k}{k} \binom{n+p-2k}{p-k} = \sum_{k=0}^{n\wedge p} \frac{(n+p-k)!}{k!(p-k)!(n-k)!}. \tag{12}$$

To see that (12) actually holds, consider that a walk in the matrix according to the above rule is a sequence of 'r' (for right), 'a' (for across) and 'd' (for down), depending on whether it goes from (i,j) to $(i,j+1)$, $(i+1,j)$ or $(i+1,j+1)$. If R denotes the number of 'r', D the number of 'd' and A the number of 'a', the constraints on A, D and R are

$$0 \leq A \leq n \wedge p, \quad 0 \leq D \leq p, \quad 0 \leq R \leq n,$$

while $A + D = p$ and $R + A = n$. This implies that $0 \leq A \leq n \wedge p$ and that R and D are determined by A. For a given k, if $A = k$, the number

of terms in the sequence is then $k + (p-k) + (n-k) = p + n - k$ and the number of differnt allocations of the 'A' steps is $\binom{k}{p+n-k}$ and, in the remaining $(p+n-k) - k = p + n - 2k$ spots, the number of different allocations of the 'R' is $\binom{p+n-2k}{n-k}$. The total number of paths is then indeed

$$\frac{(n+p-k)!}{k!(p-k)!(n-k)!}$$

for a given k.

6 Conclusions

As a coincidence, a paper by Lawrence et al. appeared in *Science* the very week of the conference. This paper is very closely related to Churchill's paper, and lays the first steps of an actual implementation of Gibbs sampling in DNA recognition. We want to mention the results contained in this paper since (a) it has been published in a wide-ranging journal and (b) it somewhat moderates the above conclusion about the practicality of Churchill's results.

Lawrence et al. (1993) deal with protein, rather than DNA, recognition, with the main difference being that the state space for a single point of the sequence is now of size 20 instead of being restricted to 4 points. In contrast to Churchill's analysis, Lawrence et al. are concerned with multiple alignment between different species, rather than reconstruction of a single sequence from fragments. The reason for this study is to exhibit a common protein structure, as long as possible, and allowing for a certain amount of discrepancy between different species. This search for common patterns is supported by an evolutionary theory of common ancestry (which we cannot describe here). The important point is that the different sequences to be compared are assumed to

be already perfectly known. In the alignment to be realized, the differences are thus explained by evolution and specificity of each species, not by chance errors in reading.

Using first a fixed size N for the almost common sequence of proteins, Lawrence et al. propose a 'Gibbs-like' algorithm which is linear in the number of sequences and in N. Their algorithm is actually closer to a stochastic perturbation of the EM algorithm, the SEM algorithm (Celeux and Diebolt, 1986; Wei and Tanner, 1990; Qian and Titterington, 1991; Robert, 1992), than to exact Gibbs sampling. To be more precise, let us mention that the authors do not simulate from the conditional posterior distribution of the parameters but rather take the corresponding expectations as current values for the next simulations of the alignment. Their generation of an alignment is not exact either, as they replace the condition posterior probability of a given alignment by its odds ratio. But their approach could be directly translated into a true Gibbs algorithm, with the addition of a distribution on the length of the common sequence.

However, in contrast to Churchill's algorithm, the reason why this algorithm works linearly, is that the authors do not allow for gaps or insertions, but only for differences in the protein sequences. The matrix of §3.3 can then be read linearly and forward. The computation times become quite reasonable since for 20 sequences of length varying from 20 to 512, convergence was attained in 1000 to 3000 iterations, (except in some cases where a suboptimal trapping state was reached).

It is thus very exciting to see variants of Gibbs sampling already implented in practice for DNA identification. Among other things, this shows an increasing

awareness of the need for more elaborate alignment methods. In addition, it may be possible to use such an approach in the warm-up steps of Churchill's algorithm, by deriving an alignment where insertions and deletions would be first omitted. More accurate MCMC algorithms could thus start from this crude alignment, which could provide a better starting point and hence lead to convergence in a reasonable time.

Finally, our overall conclusion is extremely positive. The model proposed by Churchill results in an implementable Markov chain whose output can be used to provide a valid inference about the clone sequence, including an assessment of confidence. Moreover, there is great flexibility in the underlying parameter structure which allows the model to better reflect the real process. The only drawback, if it can even be considered so, is complexity. Using data of realistic size, there are too many calculations necessary to expect usable output in our lifetimes. However, this "drawback" is a red herring for two reasons. One, with computing speed increasing every day, this algorithm soon may be computable. But second, and more important, this algorithm and model represents an exact solution to the problem. We now need to develop approximations and faster random variable generations that can be tested against this exact solution in small data sets, and are computationally feasible in realistic data sets.

Additional References

Besag, J. and Mengersen, K.L. (1993) Meta-analysis using Monte Carlo Markov Chain methods. Technical Report, Department. of Statistics, Colorado State University.

Billingsley, P. (1968) *Convergence of Probability Measures.* New York: John Wiley.

Celeux, G. and Diebolt, J. (1986) The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Comput. Statist. Quater.* **2**, 73-82.

Diebolt, J. and Robert, C.P. (1993) The Duality Principle: Discussion of Smith and Roberts, Besag and Green, and Gilks *et al.*. *J.R.S.S.* (Ser. B) **53**, 71-72.

Diebolt, J. and Robert, C.P. (1994) Estimation of finite mixture distributions by Bayesian sampling. To appear in *J.R.S.S.* (Ser. B).

Gelman, A. and Rubin, D.B. (1992) Does a single iteration suffice? In *Bayesian Statistics 4* (J.O. Berger, J.M. Bernardo, A.P. Dawid and A.F.M. Smith, eds.) London: Oxford University Press.

Karlin, S., Dembo, A., and Kawabata, T. (1990). Statistical composition of high-scoring segments from molecular sequences. *Ann. Statist. 18*, 571-581.

Lawrence, C.E., Atschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wootton, J.C. (1993) Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science* **262**, 208-214.

Muller, P. (1992) A black-box algorithm for implementing the Metropolis algorithm. Technical Report, Department of Statistics, Purdue University, Lafayette, Indiana.

Qian, W. and Titterington, D.M. (1991) Estimation of parameters in hidden Markov models. *Phil. Trans. Roy. Soc. London* A **337**, 407-428.

Robert, C.P. (1992) Discussion of Meng and Rubin In *Bayesian Statistics 4* (J.O. Berger, J.M. Bernardo, A.P. Dawid and A.F.M. Smith, eds.) London: Oxford University Press.

Robert, C.P. (1993) Convergence assesments for Monte-Carlo Markov chain methods. Technical Report, Department of Mathematics, Université de Rouen.

Tierney, L. (1991) Markov chains for exploring posterior distributions. *Computer Sciences and Statistics: Proc. 23d Symp. Interface*, 563-570.

Wei, G.C.G. and Tanner, M. (1990) A Monte Carlo implementation of the EM algorithm and the poor's man data augmentation algorithms. *J. American Statist. Assoc.* **85**, 699-704.